UNITED STATES PATENT AND TRADEMARK OFFICE

50

UNITED STATES DEPARTMENT OF COMMERCE
United States Patent and Trademark Office
Address: COMMISSIONER FOR PATENTS
P.O. Box 1450
Alexandria, Virginia 22313-1450
www.uspto.gov

APPLICATION NO.	FILING DATE	FIRST NAMED INVENTOR	ATTORNEY DOCKET NO.	CONFIRMATION NO.
10/717,671	11/21/2003	Anuj Dua	10001618-3	9068

22879 7590 06/26/2006

HEWLETT PACKARD COMPANY
P O BOX 272400, 3404 E. HARMONY ROAD
INTELLECTUAL PROPERTY ADMINISTRATION
FORT COLLINS, CO 80527-2400

EXAMINER	
HUISMAN, DAVID J	

ART UNIT	PAPER NUMBER
2183	

DATE MAILED: 06/26/2006

Please find below and/or attached an Office communication concerning this application or proceeding.

Office Action Summary	**Application No.** 10/717,671	**Applicant(s)** DUA ET AL.	
	Examiner David J. Huisman	**Art Unit** 2183	

-- The MAILING DATE of this communication appears on the cover sheet with the correspondence address --

Period for Reply

A SHORTENED STATUTORY PERIOD FOR REPLY IS SET TO EXPIRE 3 MONTH(S) OR THIRTY (30) DAYS, WHICHEVER IS LONGER, FROM THE MAILING DATE OF THIS COMMUNICATION.

- Extensions of time may be available under the provisions of 37 CFR 1.136(a). In no event, however, may a reply be timely filed after SIX (6) MONTHS from the mailing date of this communication.
- If NO period for reply is specified above, the maximum statutory period will apply and will expire SIX (6) MONTHS from the mailing date of this communication.
- Failure to reply within the set or extended period for reply will, by statute, cause the application to become ABANDONED (35 U.S.C. § 133). Any reply received by the Office later than three months after the mailing date of this communication, even if timely filed, may reduce any earned patent term adjustment. See 37 CFR 1.704(b).

Status

1)☒ Responsive to communication(s) filed on 21 November 2003.

2a)☐ This action is **FINAL**. 2b)☒ This action is non-final.

3)☐ Since this application is in condition for allowance except for formal matters, prosecution as to the merits is closed in accordance with the practice under *Ex parte Quayle*, 1935 C.D. 11, 453 O.G. 213.

Disposition of Claims

4)☒ Claim(s) 1-15 is/are pending in the application.

4a) Of the above claim(s) _____ is/are withdrawn from consideration.

5)☐ Claim(s) _____ is/are allowed.

6)☒ Claim(s) 1-15 is/are rejected.

7)☐ Claim(s) _____ is/are objected to.

8)☐ Claim(s) _____ are subject to restriction and/or election requirement.

Application Papers

9)☐ The specification is objected to by the Examiner.

10)☒ The drawing(s) filed on 21 November 2003 is/are: a)☒ accepted or b)☐ objected to by the Examiner.

Applicant may not request that any objection to the drawing(s) be held in abeyance. See 37 CFR 1.85(a).

Replacement drawing sheet(s) including the correction is required if the drawing(s) is objected to. See 37 CFR 1.121(d).

11)☐ The oath or declaration is objected to by the Examiner. Note the attached Office Action or form PTO-152.

Priority under 35 U.S.C. § 119

12)☐ Acknowledgment is made of a claim for foreign priority under 35 U.S.C. § 119(a)-(d) or (f).

a)☐ All b)☐ Some * c)☐ None of:

1.☐ Certified copies of the priority documents have been received.

2.☐ Certified copies of the priority documents have been received in Application No. _____.

3.☐ Copies of the certified copies of the priority documents have been received in this National Stage application from the International Bureau (PCT Rule 17.2(a)).

* See the attached detailed Office action for a list of the certified copies not received.

Attachment(s)

1) ☒ Notice of References Cited (PTO-892)

2) ☐ Notice of Draftsperson's Patent Drawing Review (PTO-948)

3) ☐ Information Disclosure Statement(s) (PTO-1449 or PTO/SB/08) Paper No(s)/Mail Date _____.

4) ☐ Interview Summary (PTO-413) Paper No(s)/Mail Date. _____ .

5) ☐ Notice of Informal Patent Application (PTO-152)

6) ☐ Other: _____.

DETAILED ACTION

1. Claims 1-15 have been examined.

Claim Objections

2. Claim 5 is objected to because of the following informalities: It is suggested that applicant replace the phrase "open entries on the macroinstruction queue" with --open entries in the macroinstruction queue--. Appropriate correction is required.

3. Claim 11 is objected to because of the following informalities: Please replace "memory subsystems" with --memory subsystem--. Appropriate correction is required.

4. Claim 13 is objected to because of the following informalities: In line 1, the phrase "if pending fetch request is canceled" is grammatically incorrect. Appropriate correction is required.

Double Patenting

5. The nonstatutory double patenting rejection is based on a judicially created doctrine grounded in public policy (a policy reflected in the statute) so as to prevent the unjustified or improper timewise extension of the "right to exclude" granted by a patent and to prevent possible harassment by multiple assignees. A nonstatutory obviousness-type double patenting rejection is appropriate where the conflicting claims are not identical, but at least one examined application claim is not patentably distinct from the reference claim(s) because the examined application claim is either anticipated by, or would have been obvious over, the reference claim(s). See, e.g., In re Berg, 140 F.3d 1428, 46 USPQ2d 1226 (Fed. Cir. 1998); In re Goodman, 11 F.3d 1046, 29 USPQ2d 2010 (Fed. Cir. 1993); In re Longi, 759 F.2d 887, 225 USPQ 645 (Fed. Cir. 1985); In re Van Ornum, 686 F.2d 937, 214 USPQ 761 (CCPA 1982); In re Vogel, 422 F.2d 438, 164 USPQ 619 (CCPA 1970); and In re Thorington, 418 F.2d 528, 163 USPQ 644 (CCPA 1969).

A timely filed terminal disclaimer in compliance with 37 CFR 1.321(c) or 1.321(d) may be used to overcome an actual or provisional rejection based on a nonstatutory double patenting ground provided the conflicting application or patent either is shown to be commonly owned with this application, or claims an invention made as a result of activities undertaken within the scope of a joint research agreement.

Effective January 1, 1994, a registered attorney or agent of record may sign a terminal disclaimer. A terminal disclaimer signed by the assignee must fully comply with 37 CFR 3.73(b).

6. Claims 1-3 and 9 are rejected on the ground of nonstatutory obviousness-type double patenting as being unpatentable over claim 1 of U.S. Patent No. 6,678,817 (hereafter '817) in view of Kahle et al., U.S. Patent No. 5,732,235 (herein referred to as Kahle) and Kane et al., U.S. Patent No. 5,537,559 (herein referred to as Kane).

a) Regarding claims 1 and 3, claim 1 of '817 anticipates claim 1 and claim 3 of the instant application for all but three limitations.

1) '817 has not taught sending a fetch request signal to the fetch engine and sending a fetch address to the fetch engine. However, Kahle has taught both concepts. See Fig.2, components 58 and 52. Basically, if a branch occurs, component 58 predicts the next prefetch address and then sends the address to the prefetch unit (fetch engine). By doing so, a request is also sent to the fetch engine. By having the emulation engine request fetching of its own instructions, the rest of the processor (including fetch engine) can continue doing its own thing without having to worry about requesting that fetching be done for the emulation engine when it is possible that the buffer is already full in the emulation engine. As a result, in order to increase efficiency, it would have been obvious to one of ordinary skill in the art at the time of the invention to modify '817 such that the engine of the emulated ISA requests the line of instructions and the fetch engine sends the line of instructions to the engine of the emulated ISA.

2) '817 has not taught that the fetch complete signal is a signal separate from the line of instructions. However, Kane has taught sending a fetch complete signal to an instruction buffer (Fig.4, component 480) in the form of an exception status signal. It should be noted that this

status signal arrives simultaneously with the fetched instruction, since they go hand-in-hand. See column 11, lines 60-67, of Kane. Therefore, this signal is a signal separate from the line of instructions that signifies that the fetch is complete. This exception status signal allows for a simple solution for tracking address-exceptions and generating the exceptions at the appropriate point in time (i.e., immediately upon execution). See column 3, lines 57-65, of Kane. A person of ordinary skill in the art would have recognized that exceptions should be handled in all systems (including 817's). Therefore, it would have been obvious to one of ordinary skill in the art at the time of the invention to provide such a signal to the instruction buffer of '817 which not only acts as a fetch complete signal, but also improves the efficiency of a system dealing with exceptions.

b) Regarding claim 2, claim 1 of '817 processes a plurality of fetch requests at the same time because a line of instructions is fetched and a line has multiple instructions. So, if there is one request per instruction, then multiple requests are handled at once.

c) Regarding claim 9, claim 1 of '817 has not taught sending a fetch request signal to the fetch engine along with the fetch address. However, Kane has taught sending a fetch complete signal to an instruction buffer (Fig.4, component 480) in the form of an exception status signal. It should be noted that this status signal arrives simultaneously with the fetched instruction, since they go hand-in-hand. See column 11, lines 60-67, of Kane. Therefore, this signal is a signal separate from the line of instructions that signifies that the fetch is complete. This exception status signal allows for a simple solution for tracking address-exceptions and generating the exceptions at the appropriate point in time (i.e., immediately upon execution). See column 3, lines 57-65, of Kane. A person of ordinary skill in the art would have recognized that exceptions should be handled in all systems (including 817's). Therefore, it would have been obvious to one of ordinary skill in the art at the time of the invention to provide such a signal to the instruction buffer of '817 which not only acts as a fetch complete signal, but also improves the efficiency of a system dealing with exceptions.

7. Claims 4-5 are rejected on the ground of nonstatutory obviousness-type double patenting as being unpatentable over claim 1 of '817 in view of Kahle, Kane, and Papworth et al., U.S. Patent No. 5,584,037 (herein referred to as Papworth). More specifically, while claim 1 of '817 has not taught the specifics of claims 4-5, but Papworth has taught the concept of using a speculative write pointer to track when a queue will be filled based on fetch requests. See column 2, lines 20-33. Clearly, once the queue is full, the queue will no longer accept data until some data is taken out. As a result, once the queue is full, it is inherent that fetch requests be controlled (halted), otherwise data could be incorrectly overwritten in the queue. Consequently, it would have been obvious to one of ordinary skill in the art at the time of the invention to modify '817 to include a speculative write pointer for tracking queue fullness and controlling the sending of fetch requests accordingly.

8. Claim 6 is rejected on the ground of nonstatutory obviousness-type double patenting as being unpatentable over claim 1 of '817 in view of Kahle, Kane, and Whitted III et al., U.S. Patent No. 5,515,521 (herein referred to as Whitted). More specifically, while claim 1 of '817 has not taught that if a pending fetch request is canceled due to a pipeline flush, a pending fetch request signal is canceled, and a fetch address queue is cleared, Whitted has taught this type of

procedure. See column 9, lines 54-67. A person of ordinary skill in the art would have recognized that in branch situations, either the target address or the subsequent address would be fetched next. If the target address is to be fetched next (i.e. taken branch), then all of the subsequent pending requests that are on the non-taken path do not need to be executed. Therefore, canceling the following pending requests that are unrelated to the branch along with clearing the fetch address queue would ensure that improper fetches are discarded and rollback is avoided (rollback being the term used to describe the corrections made to values that were incorrectly updated by instructions that should not have executed). Therefore, in order to avoid fetching and executing improper instructions, it would have been obvious to one of ordinary skill in the art at the time of the invention to cancel a pending fetch request and clear the fetch address queue due to a pipeline flush.

9. Claims 7-8 and 10 are rejected on the ground of nonstatutory obviousness-type double patenting as being unpatentable over claim 1 of '817 in view of Kahle and Papworth.

10. More specifically, regarding claims 7-8, while '817 has not taught sending a fetch address to a fetch engine, Kahle has taught such a concept. See Fig.2, components 58 and 52. Basically, if a branch occurs, component 58 predicts the next prefetch address and then sends the address to the prefetch unit (fetch engine). By having the emulation engine request fetching of its own instructions, the rest of the processor (including fetch engine) can continue doing its own thing without having to worry about requesting that fetching be done for the emulation engine when it is possible that the buffer is already full in the emulation engine. As a result, in order to increase efficiency, it would have been obvious to one of ordinary skill in the art at the time of

the invention to modify '817 such that the engine of the emulated ISA requests the line of instructions and the fetch engine sends the line of instructions to the engine of the emulated ISA. Furthermore, '817 has not taught determining whether the macroinstruction queue is full or will become full with one or more lines of instructions returning from one or more pending fetch requests; and performing fetching and sending of instructions if the macroinstruction queue is not full and will not become full with one or more lines of instructions returning from one or more pending fetch requests, and in addition has not taught the speculative write pointer of claim 8. However, Papworth has taught the concept of using a speculative write pointer to track when a queue will be filled based on fetch requests. See column 2, lines 20-33. Clearly, once the queue is full, the queue will no longer accept data until some data is taken out. As a result, once the queue is full, it is inherent that fetch requests be controlled (halted), otherwise data could be incorrectly overwritten in the queue. Consequently, it would have been obvious to one of ordinary skill in the art at the time of the invention to modify '817 to include a speculative write pointer for tracking queue fullness and controlling the sending of fetch requests accordingly.

11. Claims 11 and 14 are rejected on the ground of nonstatutory obviousness-type double patenting as being unpatentable over claim 1 of '817. Although the conflicting claims are not identical, they are not patentably distinct from each other because claims 11 and 14 of the instant application are anticipated by claim 1 of '817 in that claim 1 of '817 contains all of the limitations of claims 11 and 14 of the instant application. Therefore, claims 11 and 14 of the instant application are not patently distinct from the earlier patent claim and as such is unpatentable for obvious-type double patenting.

12. Claim 12 is rejected on the ground of nonstatutory obviousness-type double patenting as being unpatentable over claim 1 of '817 in view of Kahle. More specifically, while claim 1 of '817 has not taught that the engine of the emulated ISA requests the line of instructions and the fetch engine sends the line of instructions to the engine of the emulated ISA, Kahle has taught such a concept. See Fig.2, and note that the emulated engine (specifically component 58 with component 52) requests instructions and the memory will send instructions in return. Clearly, in '817, either the emulation engine or not the emulation engine (the remaining portion of the processor) will request that instructions be fetched for the emulation engine. By having the emulation engine request fetching of its own instructions, the rest of the processor (including fetch engine) can continue doing its own thing without having to worry about requesting that fetching be done for the emulation engine when it is possible that the buffer is already full in the emulation engine. As a result, in order to increase efficiency, it would have been obvious to one of ordinary skill in the art at the time of the invention to modify '817 such that the engine of the emulated ISA requests the line of instructions and the fetch engine sends the line of instructions to the engine of the emulated ISA. Furthermore, as shown in In re Karlson, 153 USPQ 184 (CCPA 1963), the elimination of an element or its function is generally not given patentable weight or would have been an obvious improvement. Consequently, it would have been obvious to one of ordinary skill in the art at the time of the invention to remove the last three items (paragraphs) of '817's claim 1 (the macroinstruction queue..., a queue in the emulation engine..., and pipeline advance logic...). And, at the very least, one motivation for removal of these items would be the processor consuming less power.

13. Claim 13 is rejected on the ground of nonstatutory obviousness-type double patenting as being unpatentable over claim 1 of '817 in view of Whitted. More specifically, while claim 1 of '817 has not taught that if a pending fetch request is canceled due to a pipeline flush, a pending fetch request signal is canceled, and a fetch address queue is cleared, Whitted has taught this type of procedure. See column 9, lines 54-67. A person of ordinary skill in the art would have recognized that in branch situations, either the target address or the subsequent address would be fetched next. If the target address is to be fetched next (i.e. taken branch), then all of the subsequent pending requests that are on the non-taken path do not need to be executed. Therefore, canceling the following pending requests that are unrelated to the branch along with clearing the fetch address queue would ensure that improper fetches are discarded and rollback is avoided (rollback being the term used to describe the corrections made to values that were incorrectly updated by instructions that should not have executed). Therefore, in order to avoid fetching and executing improper instructions, it would have been obvious to one of ordinary skill in the art at the time of the invention to cancel a pending fetch request and clear the fetch address queue due to a pipeline flush. Furthermore, as shown in In re Karlson, 153 USPQ 184 (CCPA 1963), the elimination of an element or its function is generally not given patentable weight or would have been an obvious improvement. Consequently, it would have been obvious to one of ordinary skill in the art at the time of the invention to remove the last three items (paragraphs) of '817's claim 1 (the macroinstruction queue..., a queue in the emulation engine..., and pipeline advance logic...). And, at the very least, one motivation for removal of these items would be the processor consuming less power.

14. Claim 15 is rejected on the ground of nonstatutory obviousness-type double patenting as being unpatentable over claim 1 of '817 in view of Papworth. More specifically, while claim 1 of '817 has not taught a speculative write pointer that prevents the macroinstruction queue from becoming oversubscribed by one or more pending fetch requests, wherein the speculative write pointer may be used to control the sending of a fetch request, Papworth has taught the concept of using a speculative write pointer to track when a queue will be filled based on fetch requests. See column 2, lines 20-33. Clearly, once the queue is full, the queue will no longer accept data until some data is taken out. As a result, once the queue is full, it is inherent that fetch requests be controlled (halted), otherwise data could be incorrectly overwritten in the queue. Consequently, it would have been obvious to one of ordinary skill in the art at the time of the invention to modify '817 to include a speculative write pointer for tracking queue fullness and controlling the sending of fetch requests accordingly. Furthermore, as shown in In re Karlson, 153 USPQ 184 (CCPA 1963), the elimination of an element or its function is generally not given patentable weight or would have been an obvious improvement. Consequently, it would have been obvious to one of ordinary skill in the art at the time of the invention to remove the last three items (paragraphs) of '817's claim 1 (the macroinstruction queue..., a queue in the emulation engine..., and pipeline advance logic...). And, at the very least, one motivation for removal of these items would be the processor consuming less power.

Claim Rejections - 35 USC § 103

15. The following is a quotation of 35 U.S.C. 103(a) which forms the basis for all obviousness rejections set forth in this Office action:

> (a) A patent may not be obtained though the invention is not identically disclosed or described as set forth in section 102 of this title, if the differences between the subject matter sought to be patented and the prior art are such that the subject matter as a whole would have been obvious at the time the invention was made to a person having ordinary skill in the art to which said subject matter pertains. Patentability shall not be negatived by the manner in which the invention was made.

16. Claims 11, 12, and 14 are rejected under 35 U.S.C. 103(a) as being unpatentable over Kahle in view of Kane.

17. Referring to claim 11, Kahle has taught a multi-architecture computer system capable of implementing a native instruction set architecture (ISA) and an emulated ISA comprising:

a) a memory subsystem of a native ISA. See Fig.1 and column 2, lines 33-39.

b) a fetch engine of the native ISA, said fetch engine being electrically connected to the memory subsystem of the native ISA, wherein the fetch engine accesses the memory subsystem to retrieve a line of instructions from the memory subsystem. See Fig.1, and note that instructions are fetched into instruction queue 20. In addition, it is inherent that the native fetching unit and the memory in which native instructions are stored be electrically connected together; otherwise, it could not fetch from memory (note that the fetch engine could comprise the prefetch component 52 (Fig.2) and the buses the data cache to component 36). Finally, see column 5, lines 6-9, and note that every cycle, two instructions are fetched (i.e., the fetch bandwidth is two). These two instructions make up a line of instructions, where the line is associated with the fetch address applied to the memory in that cycle.

c) an engine of an emulated ISA, wherein the engine of the emulated ISA is electrically connected to the fetch engine and interfaces with the fetch engine using a handshake protocol, wherein the engine of the emulated ISA receives a line of instructions. See Fig.2 and note that the native fetch engine is connected to the emulation engine. In general, handshaking is the general communication between two components in order to complete a task. In Kahle, the fetch engine fetches instructions and sends the guest instructions to the emulation engine 36. The emulation engine will then convert the guest instructions into native instructions and send them to the instruction queue portion of the fetch engine. See column 2, lines 57-67. So, the handshaking protocol comprises the fetch engine telling the emulation engine to take action on guest instructions and the emulation engine then tells the fetch engine that the action has been taken and that processing may continue on the converted guest instructions. And, recall that a line of instructions is received. See column 5, lines 6-9.

d) Kahle has not taught that the engine of the emulated ISA receives a fetch complete signal from the fetch engine. However, Kane has taught sending a fetch complete signal to an instruction buffer (Fig.4, component 480) in the form of an exception status signal. It should be noted that this status signal arrives simultaneously with the fetched instruction, since they go hand-in-hand. See column 11, lines 60-67, of Kane. Therefore, this signal is a signal separate from the line of instructions that signifies that the fetch is complete. This exception status signal allows for a simple solution for tracking address-exceptions and generating the exceptions at the appropriate point in time (i.e., immediately upon execution). See column 3, lines 57-65, of Kane. A person of ordinary skill in the art would have recognized that such exception signals would be applicable in Kahle's system since Kahle is concerned with address exceptions. From Fig.2 (component 54) of Kahle, it is seen that limit and attribute checks are made and exceptions are generated by the guest instructions. Therefore, it would have been obvious to one of ordinary skill in the art at the time of the invention to provide such a signal to the instruction buffer (Fig.2, component 50) in Kahle's emulation engine system which not only acts as a fetch complete signal, but also improves the efficiency of a system dealing with exceptions.

e) Kahle has not taught a fetch address queue that stores a fetch address for the line of instructions retrieved from the memory subsystem, wherein the fetch address queue is controlled by the fetch complete signal such that the fetch address is stored in the fetch address queue until the fetch complete signal is received. However, Kane has taught such a concept. See Fig.4, component 400, and column 11, lines 30-47. In this passage, Kane has disclosed that the reason fetch addresses are buffered (and postponed) is because operand (data) fetches have higher priority so that the operation of the CPU is not unnecessarily suspended by pending fetch requests. A person of ordinary skill in the art would have recognized that this concept is applicable to Kahle since emulation instructions and data are stored in the data cache (just as instructions and data are stored in the same cache within Kane's system). See Fig.1, column 2, lines 57-59, and column 3, lines 13-14, of Kahle. Since data and instructions are fetched from the data cache, then a fetch request and an operand request occurring at the same time would result in contention on the bus. As a result, it would have been obvious to one of ordinary skill in the art at the time of the invention to modify Kahle to include a fetch address queue as taught in Kane in order to store and postpone fetch requests to avoid unnecessary suspension of the CPU. Kane has further taught that the step of storing comprises storing the fetch address in the fetch address queue until the fetch complete signal is sent. It is disclosed, in column 12, lines 11-13, of Kane, that the fetch addresses are kept in the fetch address queue until the fetch request is performed. Consequently, the complete signal will signify that the fetch request has been performed (since it will arrive at the instruction buffer at the same time as the fetched instruction), and as a result, the corresponding fetch request will be removed from the queue.

18. Referring to claim 12, Kahle in view of Kane has taught a computer system as described in claim 11. Kahle has further taught that the engine of the emulated ISA requests the line of instructions and the fetch engine sends the line of instructions to the engine of the emulated ISA. See Fig.2, and note that when the emulation engine decodes a branch instruction (as shown in Fig.4 and Fig.6, for instance), the branch history table is accessed, which would in turn provide a target address fetch request (if the branch is predicted taken) to the prefetch mechanism for fetching a line of instructions. This line would then be sent from the memory subsystem to the emulation engine for execution.

19. Referring to claim 14, Kahle in view of Kane has taught a computer system as described in claim 11. Kahle has further taught a macroinstruction queue that stores the instructions that were retrieved by the fetch engine. See Fig 2, component 50.

20. Claim 13 is rejected under 35 U.S.C. 103(a) as being unpatentable over Kahle in view of Kane and further in view of Whitted.

21. Referring to claim 13, Kahle in view of Kane has taught a computer system as described in claim 11. Kahle in view of Kane has not taught that if a pending fetch request is canceled due to a pipeline flush, then a pending fetch request is canceled and a fetch address queue is cleared. However, Whitted has taught this type of procedure. See column 9, lines 54-67. A person of ordinary skill in the art would have recognized that in branch situations, either the target address or the subsequent address would be fetched next. If the target address is to be fetched next (i.e.

taken branch), then all of the subsequent pending requests that are on the non-taken path do not need to be executed. Therefore, canceling the following pending requests that are unrelated to the branch along with clearing the fetch address queue would ensure that improper fetches are discarded and rollback is avoided (rollback being the term used to describe the corrections made to values that were incorrectly updated by instructions that should not have executed). Therefore, in order to avoid fetching and executing improper instructions, it would have been obvious to one of ordinary skill in the art at the time of the invention to cancel a pending fetch request and clear the fetch address queue due to a pipeline flush.

22. Claim 15 is rejected under 35 U.S.C. 103(a) as being unpatentable over Kahle in view of Kane and further in view of Papworth.

23. Referring to claim 15, Kahle in view of Kane has taught a computer system as described in claim 14. Kahle in view of Kane has not taught the a speculative write pointer that prevents the macroinstruction queue from becoming oversubscribed by one or more pending fetch requests, wherein the speculative write pointer may be used to control the sending of a fetch request. However, Papworth has taught such the concept of using a speculative write pointer to track when a queue will be filled based on fetch requests. See column 2, lines 20-33. Clearly, once the queue is full, the queue will no longer accept data until some data is taken out. As a result, once the queue is full, it is inherent that fetch requests be controlled (halted), otherwise data could be incorrectly overwritten in the queue. Consequently, it would have been obvious to one of ordinary skill in the art at the time of the invention to modify Kahle in view of Kane to

include a speculative write pointer for tracking queue fullness and controlling the sending of fetch requests accordingly.

Conclusion

Any inquiry concerning this communication or earlier communications from the examiner should be directed to David J. Huisman whose telephone number is (571) 272-4168. The examiner can normally be reached on Monday-Friday (8:00-4:30).

If attempts to reach the examiner by telephone are unsuccessful, the examiner's supervisor, Eddie Chan can be reached on (571) 272-4162. The fax phone number for the organization where this application or proceeding is assigned is 571-273-8300.

Information regarding the status of an application may be obtained from the Patent Application Information Retrieval (PAIR) system. Status information for published applications may be obtained from either Private PAIR or Public PAIR. Status information for unpublished applications is available through Private PAIR only. For more information about the PAIR system, see http://pair-direct.uspto.gov. Should you have questions on access to the Private PAIR system, contact the Electronic Business Center (EBC) at 866-217-9197 (toll-free).

DJH
David J. Huisman
June 19, 2006

EDDIE CHAN
SUPERVISORY PATENT EXAMINER
TECHNOLOGY CENTER 2100

[illegible]